A Systems Engineering Process for Development of Federated Simulations

Robert H. Kewley, Jr.
Department of Systems Engineering
United States Military Academy
Robert.Kewley@usma.edu

Andreas Tolk
Department of Engineering Management and Systems Engineering
Old Dominion University
atolk@odu.edu

Keywords: Systems Engineering, Distributed Simulation, Federations, Model Driven Architectures

Abstract

This paper specifies a systems engineering process for the development of federated simulation models in order to support systems-of-systems analysis. The specification borrows ideas from the systems engineering domain, the simulation interoperability domain, and the software engineering domain. It has activities on the operational level to define how the real system operates, the system level to define the overall architecture of the federation, and the technical level to fully specify requirement to model developers. Use of this process allow the developers of federated simulations to separate these concerns and dialog more effectively with all of these groups.

1. INTRODUCTION

The requirement for a systems engineering approach to federation development stems from the challenges of representing systems-of-systems integration in a simulation environment. Typically, simulations exist that represent the operation of some of the subsystems. However, when these subsystems are integrated to form a new capability, a new simulation model is often useful to support decision making and optimization with respect to that capability. The federation simulation developer must typically rely on a series of techniques from other disciplines and the expertise of supporting developers to design the federation. This paper outlines a systems engineering approach that may be used to guide the process.

2. ENGINEERING SKILLS FOR FEDERATED SIMULATIONS

This paper proposes an systems engineering process to help orchestrate the engineering tasks required for federated simulation [1]. These federates must be held together, conceptually and technically, by the engineering capabilities:

Information Exchange System. The capability to pass meaningful information between federates during the simulation run.

Environmental Representation. The capability for federates to reference a shared and correlated environment in which entities interact.

Entity Representation. The capability for federates to reference shared conceptually aligned information about entities in the simulation. Some of this representation is passed via the information exchange system.

Models. Within the context of the analysis or training question, the internal models of each federate must be validated and coordinated across the federation.

Data Collection. The capability to collect meaningful information from the simulation run in the context of the analysis question or training objective for which the federation was designed.

3. SUPPORTING SPECIFICATIONS

The IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP) is an existing process for federation development [2]. However, this process focuses primarily on the federation object model development within the information exchange system. It fails to address coordination of the other required engineering tasks. A more comprehensive systems engineering process has been proposed in [3]. It calls for a common definition of requirements via the Military Missions and Means framework [4]. It then goes on to call for further process of information exchange system development and analysis support using Model Driven Architectures [5]. However, that process assumed the federation supported military systems analysis. This paper extends that work to further specify a more general and more formal systems engineering process that supports development of federated simulations.

Building federations requires more than a simple technical understanding of how simulations exchange data. It requires a common shared conceptual understanding of the simulation environment, entities in the models, and exchanges between them. It is very difficult to gain this by simply looking at source code and conforming to technical standards. Levels of interoperability shed some light on this challenge [6]. For example, technical interoperability is a very specific set of protocols that clearly define the standards. Conceptual interoperability, on the highest level, is a loosely defined by

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JAN 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Systems Engineering Process for Development of Federated Simulations	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **United States Military Academy,Department of Systems Engineering,West Point,NY, 10996**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Live-Virtual Constructive Conference, 12-15 Jan 2009, El Paso, TX

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**39**	

shared concept that provides context and common organizational uses for the models. For composable models, the development team must have this shared conceptual model prior to detailed engineering of the federation.

Fortunately, the software engineering community has defined a framework to support this level of interoperability. The Object Management Group's Model Driven Architecture (MDA) is an open standard that enables an organization to specify their domain expertise in a modeling language that is independent of the technology used to implement that logic [5]. This specification achieves the technical goal of abstracting the domain logic away from the technical implementation details. For a simulation model, this supports validation of the model by domain experts to enable composability.

The underlying idea is to separate business and application logic from underlying technology. To enable this, MDA defines artifacts based on the Unified Modeling Language (UML) to describe a hierarchy of models that cope with the various challenges on different levels. Guidelines for the use of MDA establish three different modeling viewpoints [7], and these can be interpreted for the simulation domain. The highest level of abstraction is the Computer Independent Models (CIM) which focus on the system's business logic. The Platform Independent Models (PIM) capture concepts and processes in software engineering artifacts. If this conceptual model is mapped to a concrete platform and implementing architecture, the result is a Platform Specific Model (PSM). In the optimal case, the PSM can be used to produce code, as all information needed is available.

MDA has the additional advantage of standardized metamodels. The Meta-Object Facility (MOF)[8] and XML Metadata Interchange (XMI)[9] declare abstractions for the representation and exchange of models. These features of MDA, if applied for modeling and simulation, allow simulation system developers to take advantage of the numerous modeling and development tools that are available in the commercial and open source community based on these standards.

4. SYSTEMS ENGINEERING PROCESS FOR DEVELOPMENT OF FEDERATED SIMULATIONS

The real challenge of simulation development is to ensure that the final product is consistent with the purposes for which the simulation project was originally started. It should support analysis of different strategies, represented as simulation inputs. In so doing, there must be a mechanism for tying different simulation development activities to the requirements for certain system functions to be modeled and to certain outputs to be produced. A systems engineering approach to simulation development ensures that these ties to requirements are maintained throughout the process.

The process borrows from the Model Driven Architectures approach to produce models of the simulation system on three different levels —targeting three different sets of stakeholders. Operational level activities produce an operational description of the system to be simulated. The operational products, analogous to the CIM of MDA, are independent of the fact that a simulation model is being built. They consider the concerns of the operational stakeholders who must use or operate the system as it functions in the real world. The system level activities focus on the simulation architecture as a whole. These products, analogous to the PIM of MDA, assign simulation functions and metrics to different simulation models. The primary stakeholder for system level activities are integration engineers and managers for each of the component models. The technical level activities focus on the detailed development of simulation components and the interfaces between them. These products, analogous to the PSM of MDA, provide sufficient detail for software engineers to develop code that will interface with the overall simulation federation to provide the required results. In some cases, the software or test cases may be auto-generated from the technical specification.

4.1. Operational Level Activities

In performing the operational level activities, simulation engineers are focused on the problem definition phase of the systems engineering process [10]. Their primary goal during this phase is to gain an understanding, documented as a combination of UML specifications and other diagrams, of the system to be modeled. They should understand its users, its functions, and the value derived from the operation of the system itself. The steps in Figure 5 represent the steps of this process.

Identify system use-cases. Identify how the system is used by different classes of users to perform the function for which it was designed. This results in a high level UML use-case model for the system.

Functionally define the system. Use the use-case model and work with system users to define in a hierarchy the functions of interest for the system. This results in a functional hierarchy for the system. Figure 2 shows this for the ground soldier command and control system.

Identify stakeholders. Identify stakeholders for the system. These are not only users, but also system owners, system developers, and the client for which the simulation study is being performed. Ensure the modeling detail captures enough information to answer the question at hand without incorporating so much detail that simulation development becomes overly difficult, expensive, and time consuming.

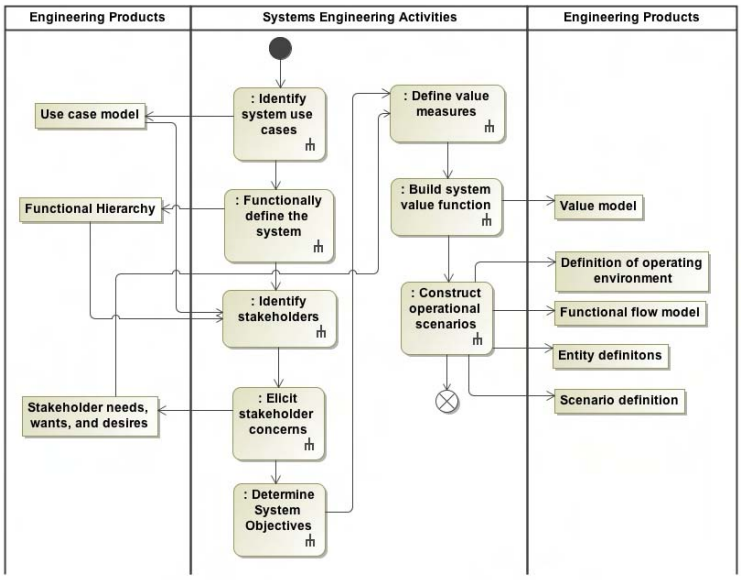


Figure 1: Operational level activities.

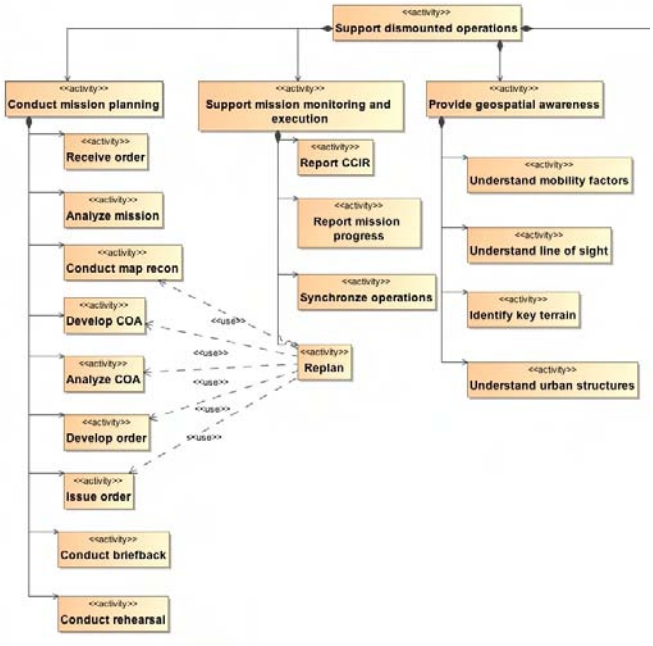


Figure 2: Functional decomposition for ground soldier command and control system.

Elicit stakeholder concerns. Using any combination of interviews, focus groups, and surveys, determine the fundamental concerns and values of the stakeholders identified in the previous steps.

Determine system objectives. Based on the functional analysis and stakeholder concerns, determine the system objectives that must be successfully met in order to deliver value back to the stakeholders.

Define value measures. Once the objectives have been identified, determine value measures that can be used to see if the system has indeed met those objectives. The simulation system, once developed, must be able to estimate system performance in terms of these objectives so that the relative performance of different alternatives can be determined. The results of this phase may be represented as a value hierarchy specified as a UML class diagram where each value measure is a component of an individual objective, and individual objectives are components of the overall system objective. It is also helpful to specify the range of possible values, from least desirable to most desirable, for each performance measure [10]. Figure 3 shows a portion of the value hierarchy for a ground soldier command and control system.

Build system value function. A simple value hierarchy is not sufficient for direct comparison between alternatives. The development team must return to the stakeholders and determine the value curves and weights for each performance measure, taking into consideration the importance and overall variability of each measure [10]. This results in a value function that translates a set of performance scores for each alternative into an overall value score that represents the value of that alternative to the system stakeholders.

Construct operational scenarios. Once the system, its

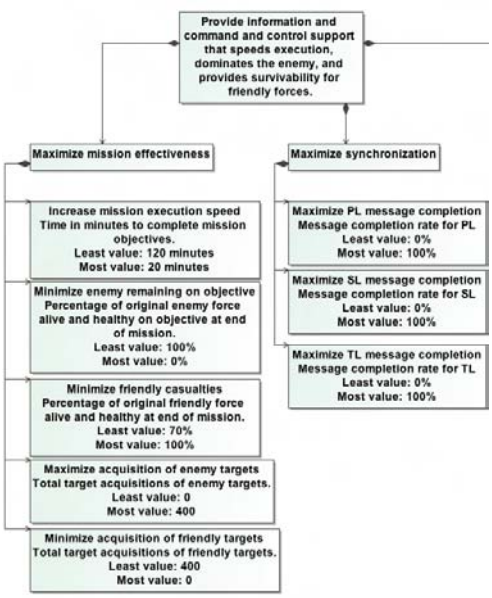


Figure 3: Value hierarchy for ground soldier command and control system.

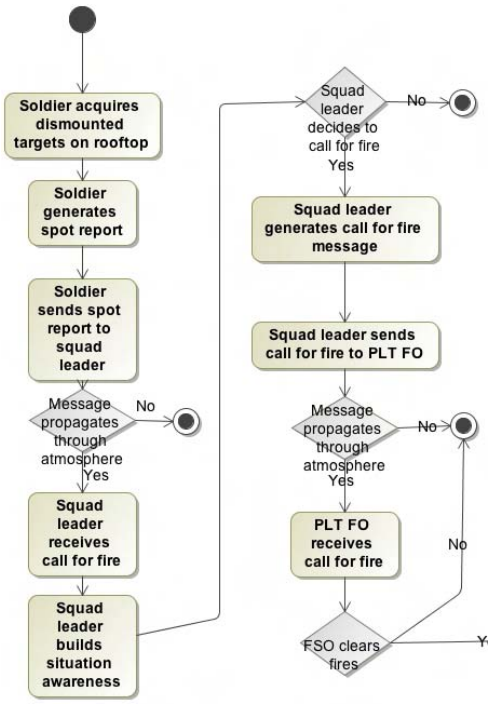


Figure 4: Functional flow model for request artillery fires function of ground soldier command and control system.

functions, and its values have been defined, the simulation study team must also define the scenario that represents the context for evaluation of system performance. In a military simulation, this represents details such as terrain and weather, forces in the engagement, supporting friendly forces, and full definitions of the entities in the simulation. The scenario definition describes the mission, forces involved, and roles of the simulated entities. In a military context, the Military Scenario Definition Language is an excellent standard for this representation [11].

Entity definitions are an important aspect of scenario definition. All too often, the names of entities can lead to ambiguous understandings of the actual capabilities represented. Entity definitions should be as specific as possible with references to authoritative sources that provide accurate data to simulation modelers who must represent these entities in their respective simulations.

Finally, within the scenario, the functions performed by the system under study should be defined as a functional flow model so that simulated events can be synchronized in the proper order. This model can be represented as a UML activity diagram. A portion of the function flow model for the artillery fires request function of the ground soldier command and control system is shown in Figure 4.

4.2. Systems Level Activities

Once the operational picture is well understood and documented, it is time for the high level systems design of the federated simulation. The design steps in this phase build UML specifications of the simulation functions, simulation data collection, information exchanges, and environmental data. These steps result in a logical allocation of functionality and information exchanges that derive from the operational requirements from the previous step. While not sufficient for writing code, these models allow simulation engineers and software engineers from the participating federates to allocate high level tasks and work packages to support simulation development.

Select participating simulations. Once the required functionality is known, the simulation engineers must research candidate federates for inclusion in the federation. Some of the considerations for selection are the capability to model desired phenomena, ease of integration with the other models, and difficulty of required modifications. In some cases, a new federate must be developed in order to model some aspects of the system.

Allocate simulation activities to specific simulation models. Once the candidate federates are selected, modeling functions must be allocated to individual federates.

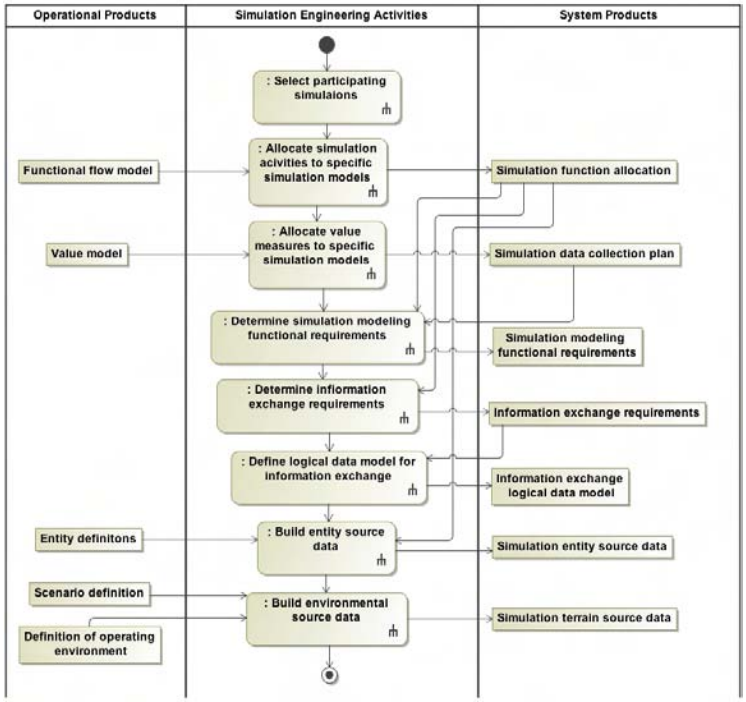


Figure 5: System level activities.

The resulting functional allocation takes the functional flow diagram from the operational level and allocates specific functions to federates using swim lanes in the UML activity diagram. A portion of this allocation for the ground soldier command and control simulation is shown in Figure 6.

Allocate value measures to specific simulation models. In a manner similar to the allocation of functions, the requirements to collect necessary performance data should be allocated to federates as well. In some cases, the required data may not exist in any one federate, but will have to be collected from network interaction data collected by loggers.

Define simulation modeling functional requirements. Once the modeling functions and data collection functions have been determined, the simulation functionality requirements may be formally specified for the models. These requirements documents may be used to support contracting with federate developers who must deliver models with the required functions.

Determine information exchange requirements. In order for the federation to execute, data must be exchanged between the models. These requirements may be derived from the activity diagrams used to allocate functions to individual federates. Any time that a control line crosses a swimlane, there is typically a requirement for some amount of information to be passed in order to support that allocation.

Define logical data model for information exchange. As information exchange requirements are identified in the previous step, engineers must formally specify the data elements required to support that data exchange. These data requirements can often be specified in a UML class diagram. This is a two-way process. It may be more efficient to delay this formal specification until the information exchange architecture is selected in the technical view. In some cases, information elements from that architecture may be reverse engineered to provide the require information elements.

Build entity source data. In addition to developing simulation software, the team must also consider the entities that will participate in the scenario. In some cases, these entities must be constructed from a significant amount of data. This step represents the collection of accurate and appropriate source data for the entities in the scenario.

Build environmental source data In addition to entities, the environment must be considered as well. This step represents the collection of source data necessary to appropriately represent the environment in the different federates. The environmental representation may not

be the same for all federates. However, using the same source data will lead to correlated representations across the models.

Once the system has been designed and data has been collected, it is still necessary to do system development for all of the participating federates and for the overall federation integration architecture. There are all technical level activities that look to provide software engineers and programmers sufficient information that will allow them to write code and deliver working federates within the overall specification. Figure 7 shows a diagram of the technical level activities and products.

Select information exchange technical architecture. The simulation must exchange information across an architecture designed for this purpose. Simulation standards such as Distributed Interactive Simulation (DIS) or the High Level Architecture (HLA) are possible choices. Another possibility is to use service oriented architectures, based on web standards, designed to support business or industrial systems.

Develop information exchange data model. The information exchange data model must be specified and represented in technical format selected in the previous step. In the case of HLA, this specification will be a federation object model (FOM). A web services architecture would require extensible markup language (XML) representations of the data. Once the information exchange data model is developed, a UML sequence diagram can be used to translate simulation functions into sequence diagrams that explicitly show the communications between federates and the simulation functions performed by each. Figure 8 shows a sequence diagram for the communication function of the ground soldier command and control federation.

Specify simulation models. Required simulation functions were determined as a systems level activity. Now these function must be specified using the language and format of the information exchange architecture. For example, in HLA, certain simulation functions could be started upon receipt of specific interactions from the runtime infrastructure (RTI). In a web services integration, these functions could be represented using the web services definition language (WSDL).

Build entity data models in simulation specific formats. In this step, the entity data collected in the system level activity must be converted into input formats that can be read by the participating federates. These representations may be databases, spreadsheets, XML files, or other file formats required by the participating

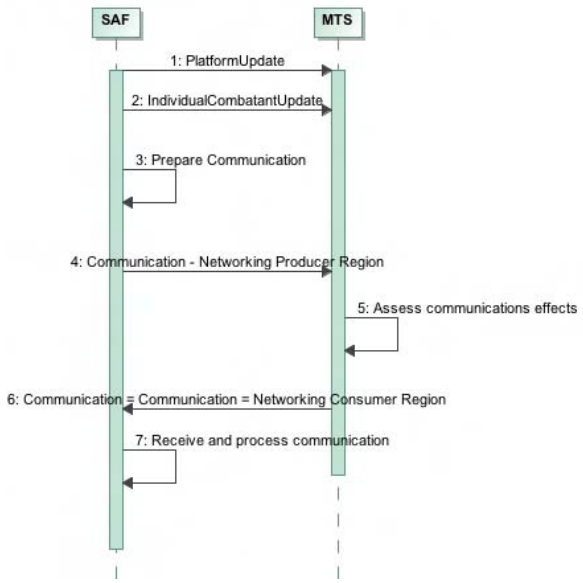


Figure 8: Sequence diagram for simulation functions and information exchanges required to support the communicate function of the ground soldier command and control simulation.

simulations. In some cases, supporting tools for the simulation can ease this transition. In other cases, it is a laborious manual process using basic editors.

Build data collection models in simulation specific formats. In this step, the data collection requirements determined in the system level activity must be represented as output data from simulation federates or from data loggers tied to the federation. Depending upon the federate, these format may be supported by standard database systems, or they may simply be text or log files that must be processed. The developers must build queries or tools that collect this raw data and summarize it as value measures from the value model built in the operational level.

Build terrain data models in simulation specific formats The source terrain data must also be converted into simulation specific formats. In many cases commercial tools support this process for a variety of formats. In other cases, the terrain data may be read into the simulation models in its existing geospatial format. The result of this step is correlated representation of the terrain across the federation.

4.3. System Development and Testing

Delivery of the engineering products described at each level will give system developers all of the specifications they

need to build software components that deliver the required functionality and interface with the federation architecture. The operational level will give them a conceptual context for integration. The system level will give them a semantic view of their components and an understanding of the overall simulation architecture. Finally the technical products will specify their components in sufficient detail to allow them to interface with the selected technical infrastructure. A good systems engineering process requires stakeholders from all three of these levels for the to work together in a coordinated way.

Despite the best intentions of a well designed architecture, there is no substitute for component and integration testing to ensure all of the pieces work as advertised. Component tests are designed around individual components and their interfaces. They typically test a discrete function by replicating the inputs from the federation and reading the outputs from the federate. The ground soldier simulation system used the Modeling Architecture for Technology, Research and Experimentation (MATREX) [12] environment. This environment supported automatic test case generation to support integration. Larger scale integration tests bring together a number of federates and test the ability of the federation to model system-level capabilities and to collect system-level data.

5. ADVANTAGES OF A SYSTEMS ENGINEERING APPROACH

There are three main advantages to using a systems engineering approach to federated simulation development. The first advantage is to ensure a clear line of logic from operational representations, to system level federation design, to coding and development. A second advantage is the separation of concerns permitted by modeling the system on three different levels. Operational experts do not have to read computer code to adjust models on the operational level. System level experts can organize and specify the system using systems architecture tools, and code developers can work from technical level specifications. The final advantage is that all of the systems engineering products support the engineering manager in implementation of the development and test plan. It breaks the complex federation into discrete pieces of functionality that can be developed, component tested, and integration tested in order to manage progress. This approach has helped during implementation of the ground soldier command and control federation, saving a great deal of development tim and effort that is typically spent rectifying poorly specified interfaces during integration tests.

REFERENCES

[1] R. H. Kewley, N. Goerger, E. Teague, D. Henderson, and J. Cook, "Federated simulations for systems of systems integration," in *Draft submitted to Proceedings of the 2008 Winter Simulation Conference*, 2008.

[2] IEEE Computer Society, "IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)," Tech. Rep. 1516.3, 2003.

[3] A. Tolk, T. Litwin, and R. Kewley, "A systems engineering process for driving federated model development with operational requirements," in *Draft submitted to Proceedings of the 2008 Winter Simulation Conference*, S. J. Mason, R. R. Hill, L. Moench., and O. Rose, Eds., December 2008.

[4] J. H. Sheehan, P. H. Dietz, B. E. Bray, B. A. Harris, and A. B. Wong, "The military missions and means framework," Army Material Systems Analysis Activity, Tech. Rep. TR-756, 2004.

[5] Object Management Group. (2007) OMG model driven architecture. http://www.omg.org/mda. accessed 14 march 2008. [Online]. Available: http://www.omg.org/mda

[6] A. Tolk, C. D. Turnitsa, S. Y. Diallo, and L. S. Winters, "Composable M&S web services for net-centric applications," *Jornal of Defense Modeling and Simulation*, vol. 3, no. 1, pp. 27–44, 2006.

[7] Object Management Group, "MDA guide version 1.0.1," Object Management Group, Tech. Rep., 2003. [Online]. Available: http://www.omg.org/technology/documents/vault.htm#modeling

[8] ——, "Meta Object Facility specification," Object Management Group, Tech. Rep. Version 1.4, 2002. [Online]. Available: http://www.omg.org/technology/documents/formal/mof.htm

[9] ——, "XMI version 1 production of XML schema specification," Object Management Group, Tech. Rep. Version 1.3, 2003. [Online]. Available: http://www.omg.org/technology/documents/vault.htm#modeling

[10] G. Parnell, P. Driscoll, and D. Henderson, Eds., *Decision Making in Systems Engineering and Management*. Hoboken: Wiley, 2008.

[11] "Military scenario definition language," Simulation Interoperability Standards Organization, 2008. [Online]. Available: http://www.sisostds.org/index.php?tg=articles&idx=More&article=440&topics=103

[12] T. Hurt, T. McKelvey, and J. McDonnell, "The modeling architecture for technology, research, and experimentation," in *Proceedings of the 2006 Winter Simulation Conference*, L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, Eds., 2006, pp. 1261–1265.

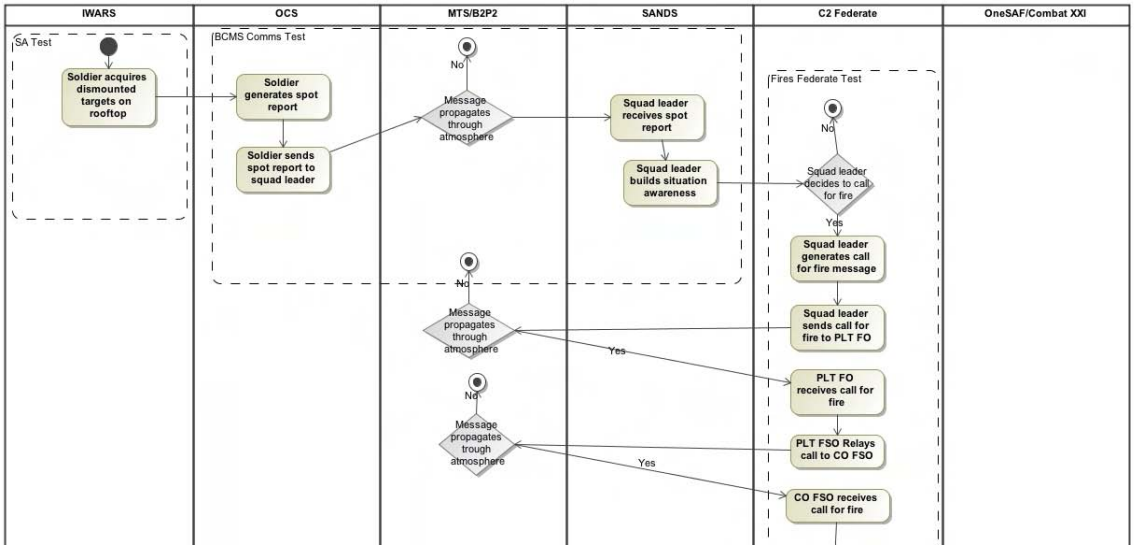


Figure 6: Allocation to request fires functions to simulation federates, represented as vertical swim lanes.

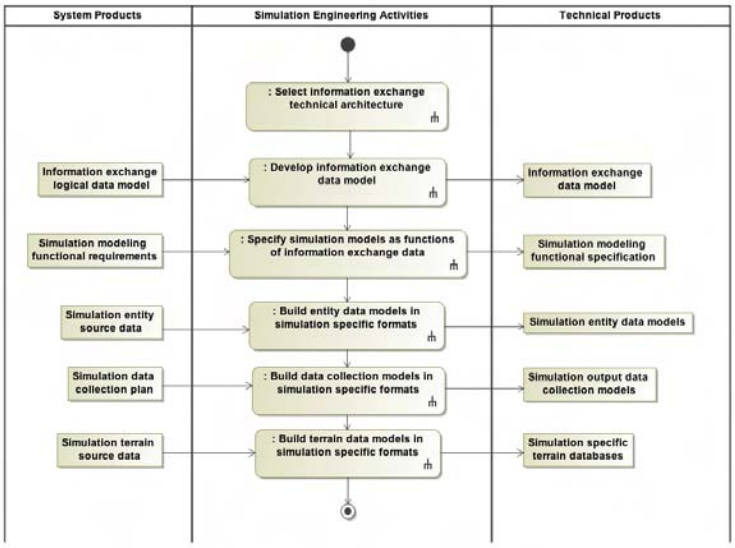


Figure 7: Technical level activities.

A Systems Engineering Approach to Developing Federated Simulations for Systems Testing

ITEA Conference, 14 January 2008

LTC Rob Kewley
Director, Operations Research Center
United States Military Academy
Department of Systems Engineering

Agenda

- Systems of systems analysis
- SysHub concept
- Engineering skills
  - Information exchange system
  - Environmental representation
  - Entity representation
  - Models
  - Data collection
  - Time management
- Systems engineering process
  - Operational View Activities
  - Systems View Activities
  - Technical View Activities
- Applications
  - Soldier as a System
  - UAS/Apache Cooperative Engagements
  - Swarming UAS

Cooperative Engagements
System of Systems

Joint Systems and Requirements

Supplemental Budget

Cooperative Engagements with ERMP and AH-64D

Institutional Bias

Current Operating Environment

Maneuver Bde TOC

GCS
MI Co
Maneuver Bde
Cav Sqdn

Program Budget

Ongoing Analysis

TEACHING FUTURE ARMY LEADERS TO SOLVE COMPLEX PROBLEMS

System of Systems Issues

"Systems of systems exist when there is a presence of a majority of the following five characteristics: operational and managerial independence, geographical distribution, emergent behavior, and evolutionary development." *[Sage and Cuppan 2001]*

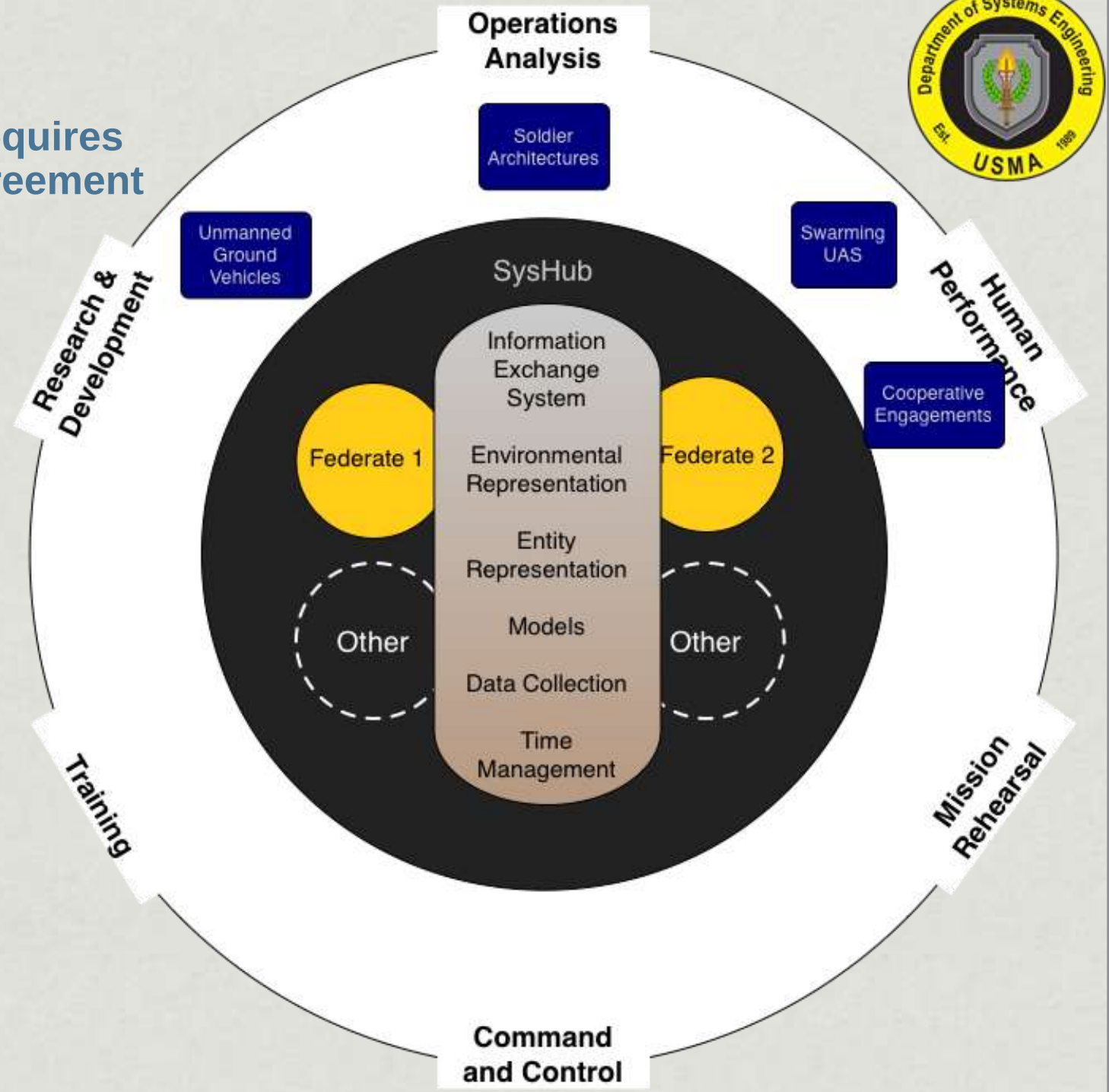
- Air space management and safety achieved by all component systems working together
- Individual systems provide useful services (fire power, real time video)
- Individual systems are acquired independently with different contractors
- Individual systems come and go routinely
- Dependent on networked communications
- Standard protocols and interfaces
- Geographically dispersed
- System complexity leads to emergent behavior
- Extensive coordination is central to achieving high levels of efficiency and safety

Why Federate?

- Analysis or performance measurement for overarching system measures
- Considering alternative sub-systems or architectures
- Sub-system models available for reuse
- Models are geographically distributed
- Building a single model of a large complex system is difficult and costly

SysHub

Composability requires more than an agreement to use HLA



TEACHING FUTURE ARMY LEADERS TO SOLVE COMPLEX PROBLEMS

Information Exchange System

The capability to pass meaningful information between federates during the simulation run.

- Levels of conceptual interoperability (Turnitsa and Tolk)
- Interoperability standards
  - Distributed Interactive Simulation
  - High Level Architecture
  - Web Services
- Interoperability tools
  - UML modeling
  - Data modeling
  - Army's Modeling Architectures for Technology, Research and Experimentation (MATREX)

Environmental Representation

The capability for federates to reference a shared and correlated environment in which entities interact.

- Reference a conceptually aligned common environment with detail appropriate for federate
- Environmental standards
  - Synthetic Environment Data Representation and Interchange Specification (SEDRIS)
  - GIS standards
- Strategies
  - One common environmental database
  - Correlated models built using a single tool
  - Correlated models build using different tools

Entity Representation

The capability for federates to referenced shared conceptually aligned information about entities in the simulation. Some of this representation is passed via the information exchange system.

- Characteristics that do not change during the simulation run
  - Data typically loaded from federate specific data sets
  - Data from different federates must be conceptually aligned with an authoritative and shared source, such as AMSAA data
- Dynamic characteristics that make up the entity state
  - State is dynamically updated based on simulation events
  - Federates access entity states via the information exchange system

Models

Within the context of the analysis or training question, the internal models of each federate must be validated and coordinated across the federation.

- Models running in a new context
- Model must be appropriate for simulation
- Example – High energy laser modeling to defeat incoming projectile
- Model validation steps
  - Verify model assumptions within the new context
  - Run the federation with data for which the results are known or can be expected to fall within a certain range
  - System experts can view simulation runs and provide face validity

Data Collection

The capability to collect meaningful information from the simulation run in the context of the analysis question or training objective for which the federation was designed.

- What is the question?
- Define question in terms of simulation inputs and outputs
- Data collection sources
  - Data collection tools for individual federates
  - Data loggers for the federation
  - Custom data collection and analysis tools for a specific federation
  - Human observation and measurement

Time Management

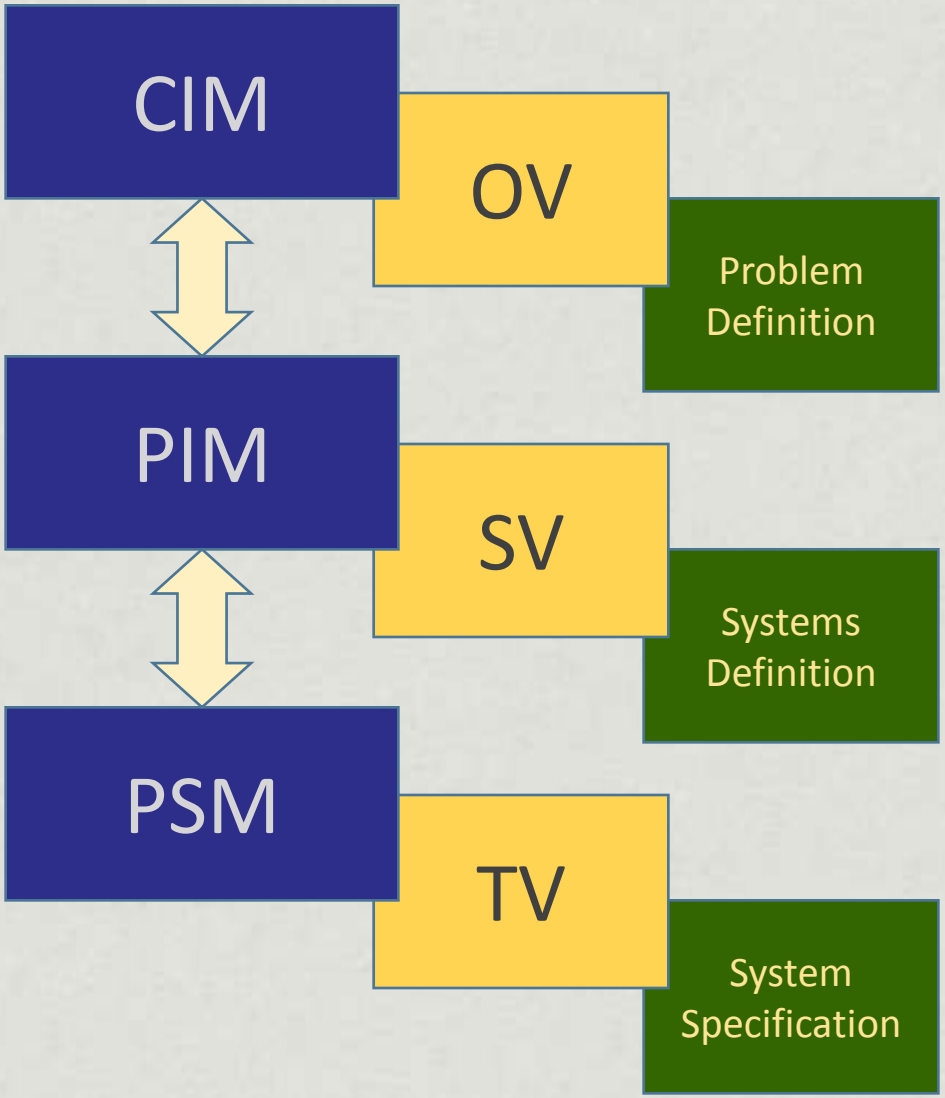
The capability to execute the simulation in separate federates in a synchronized way so that the ordering of events and interactions between entities mirror the real system.

- Simulations are synchronous
- Federates must have a shared representation of time
- Synchronization strategies
  - Execute in real time
  - Use a multiple of real time
  - Federation maintains a clock and look-ahead window
  - Federation maintains the event list

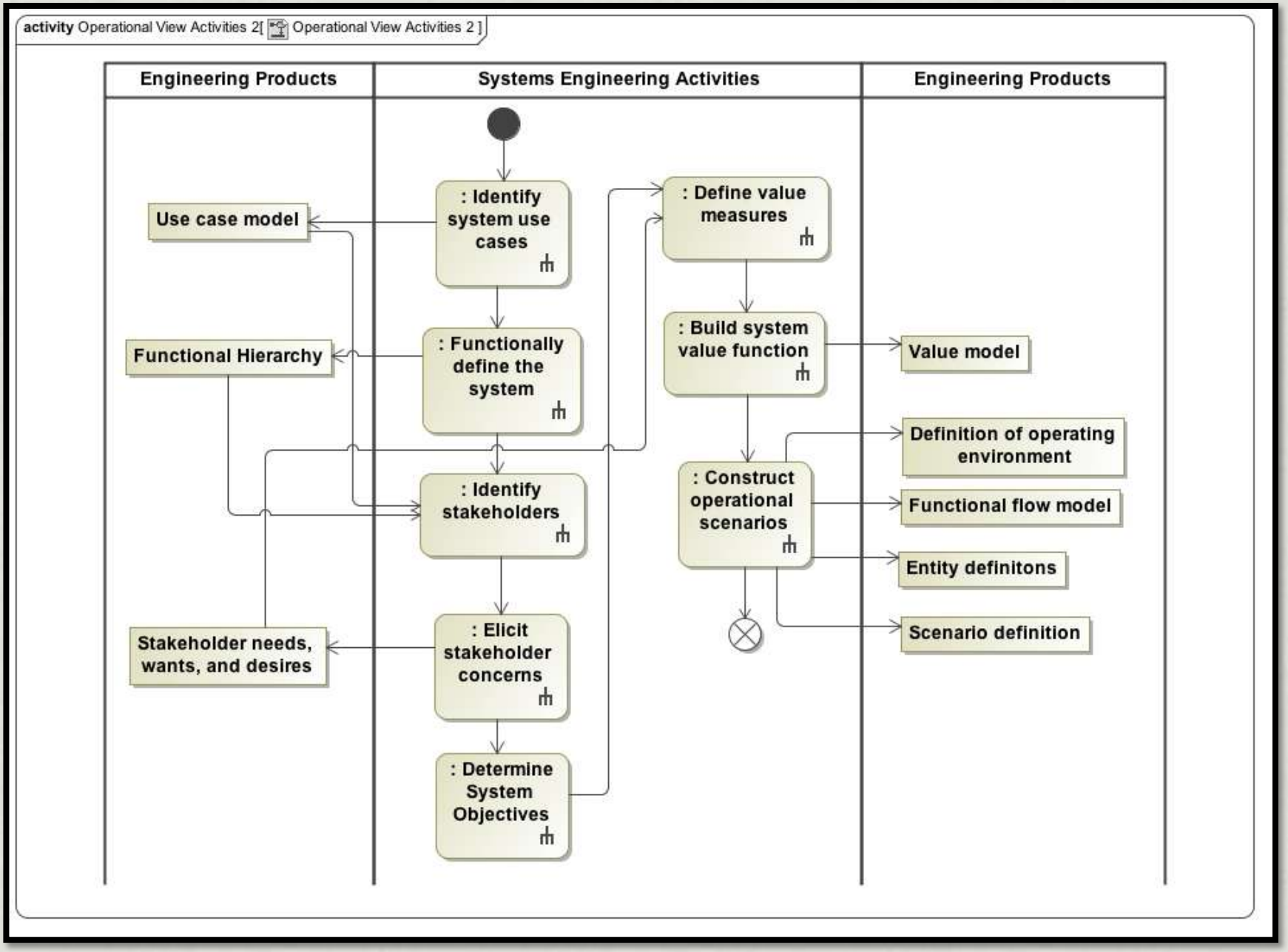
Model Driven Architectures

Similar system engineering view like captured in DoDAF (OV, SV, TV) and as supported by the Military Missions and Means Framework (MMF)

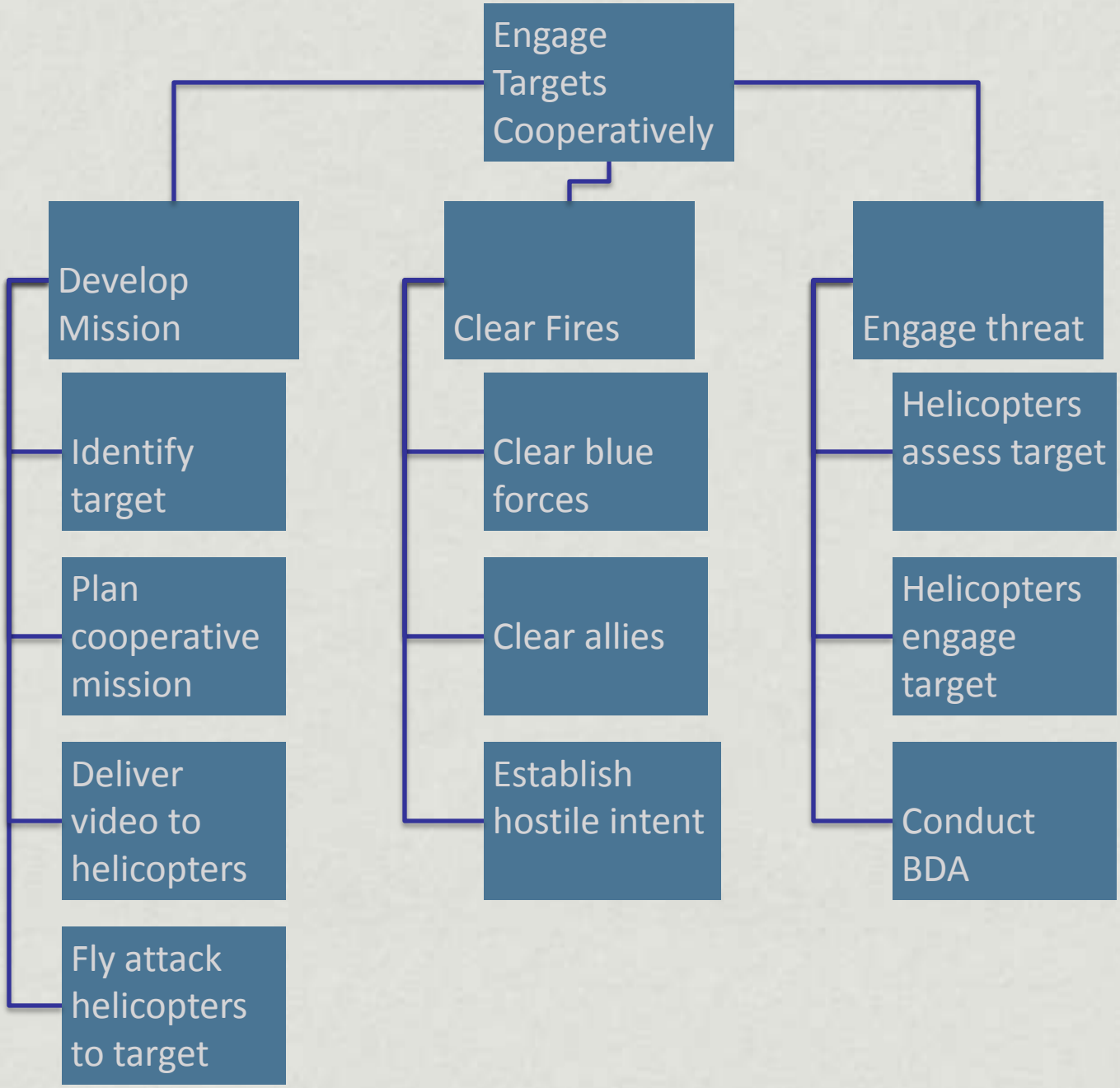
- Computer Independent Model (CIM)
  - Operational Idea, high-level view
- Platform Independent Model (PIM)
  - Algorithmic Solutions
  - Identifies classes, associations, etc.
- Platform Specific Model (PSM)
  - Code-level solution
  - Platform and language specific
- All models are connected via transformation patterns



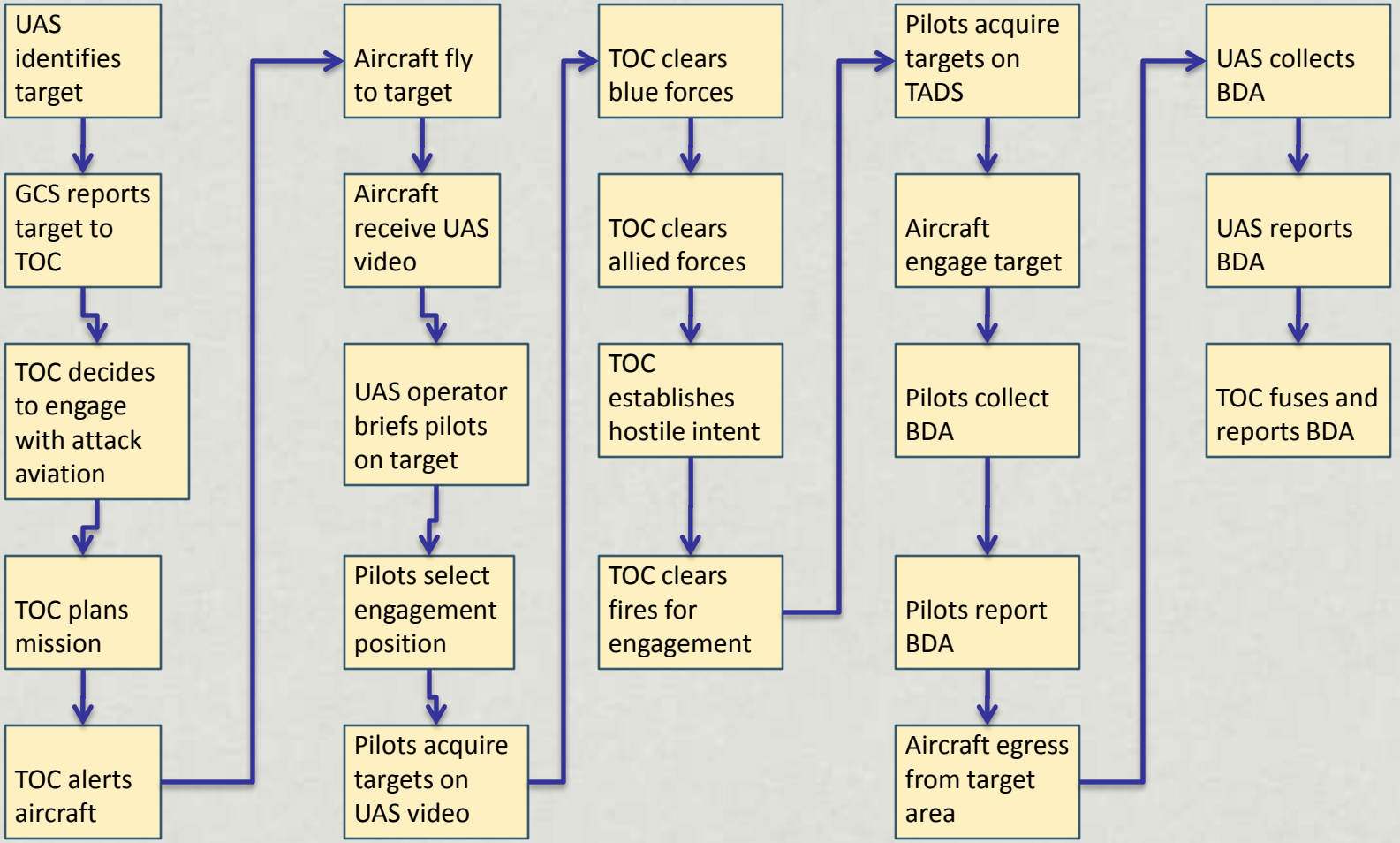
Systems Engineering Process
Operational View Activities



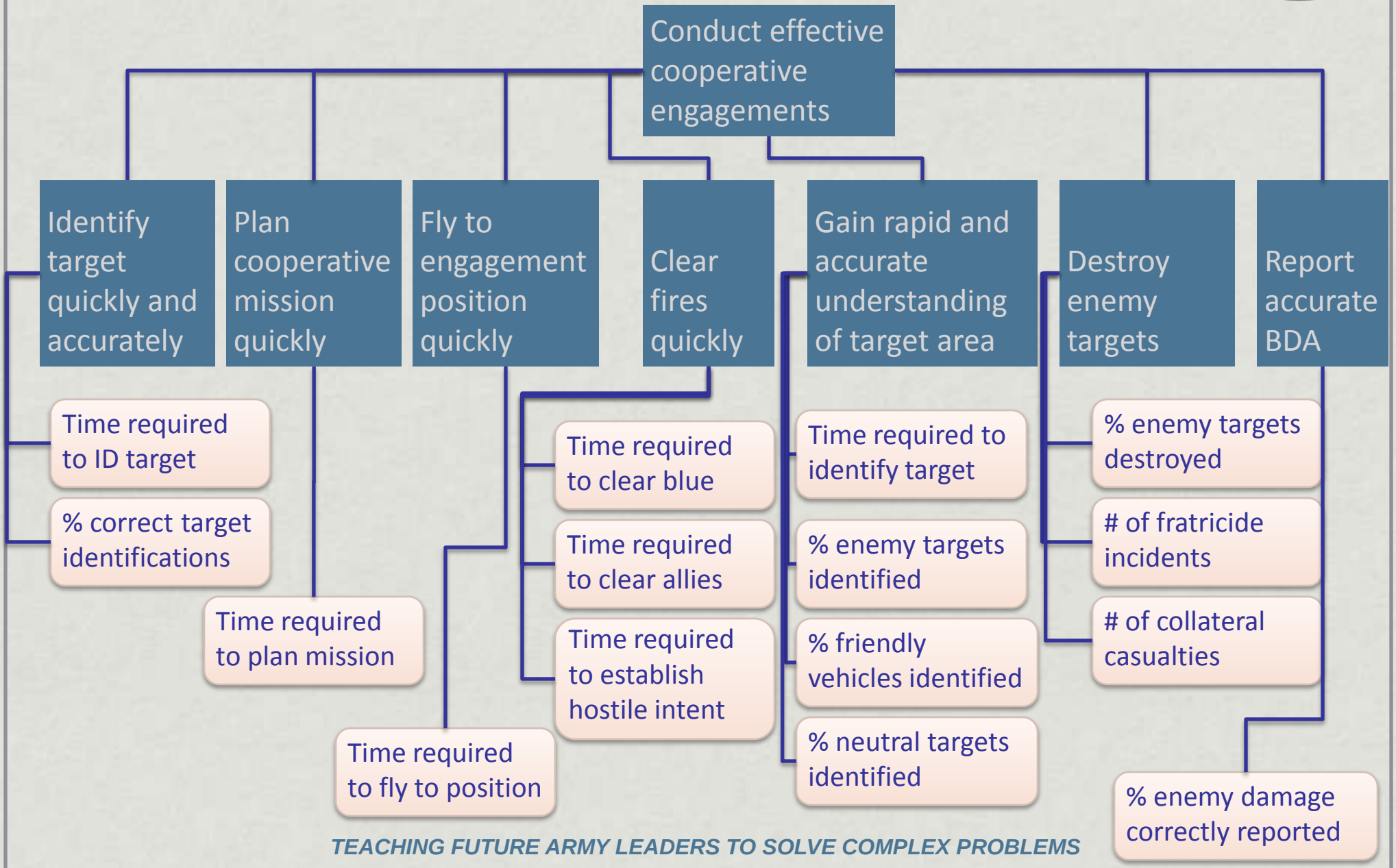
Functional Hierarchy



Mission Thread

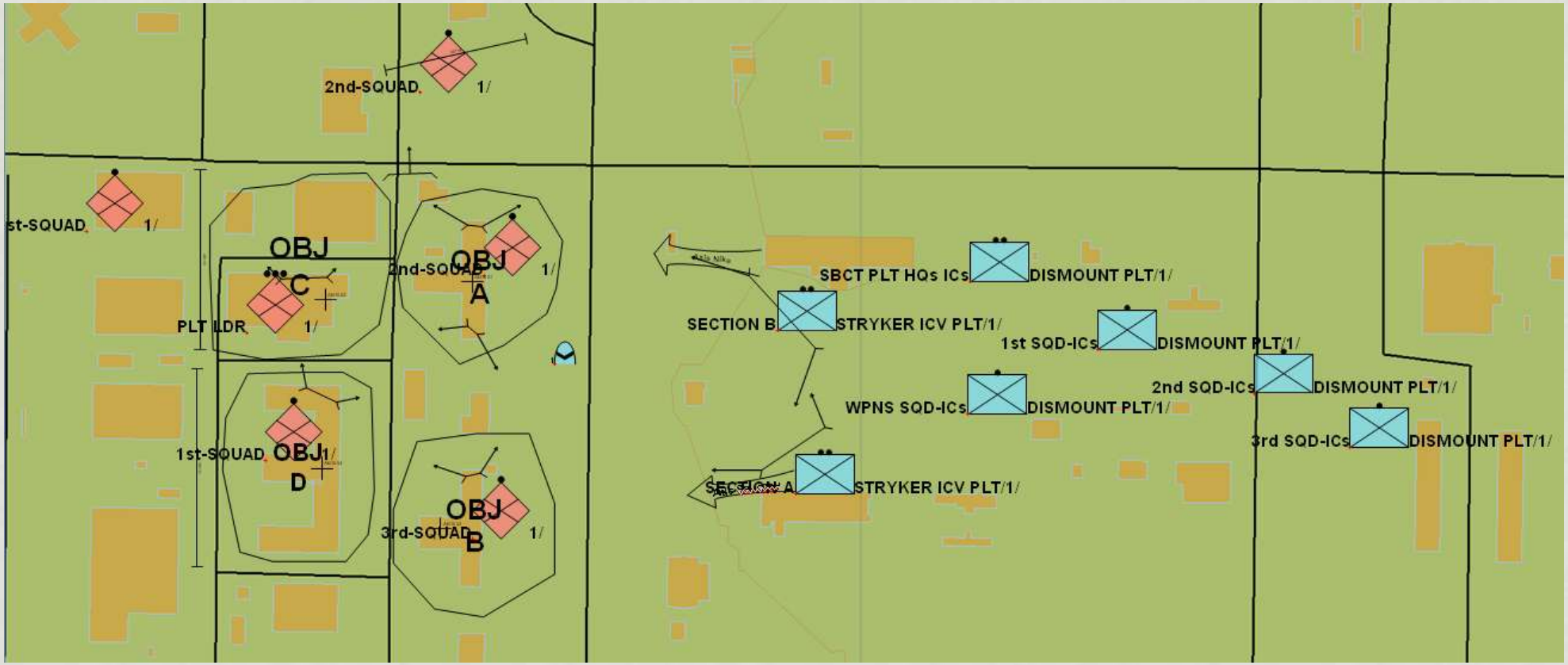


Cooperative Engagement Effectiveness

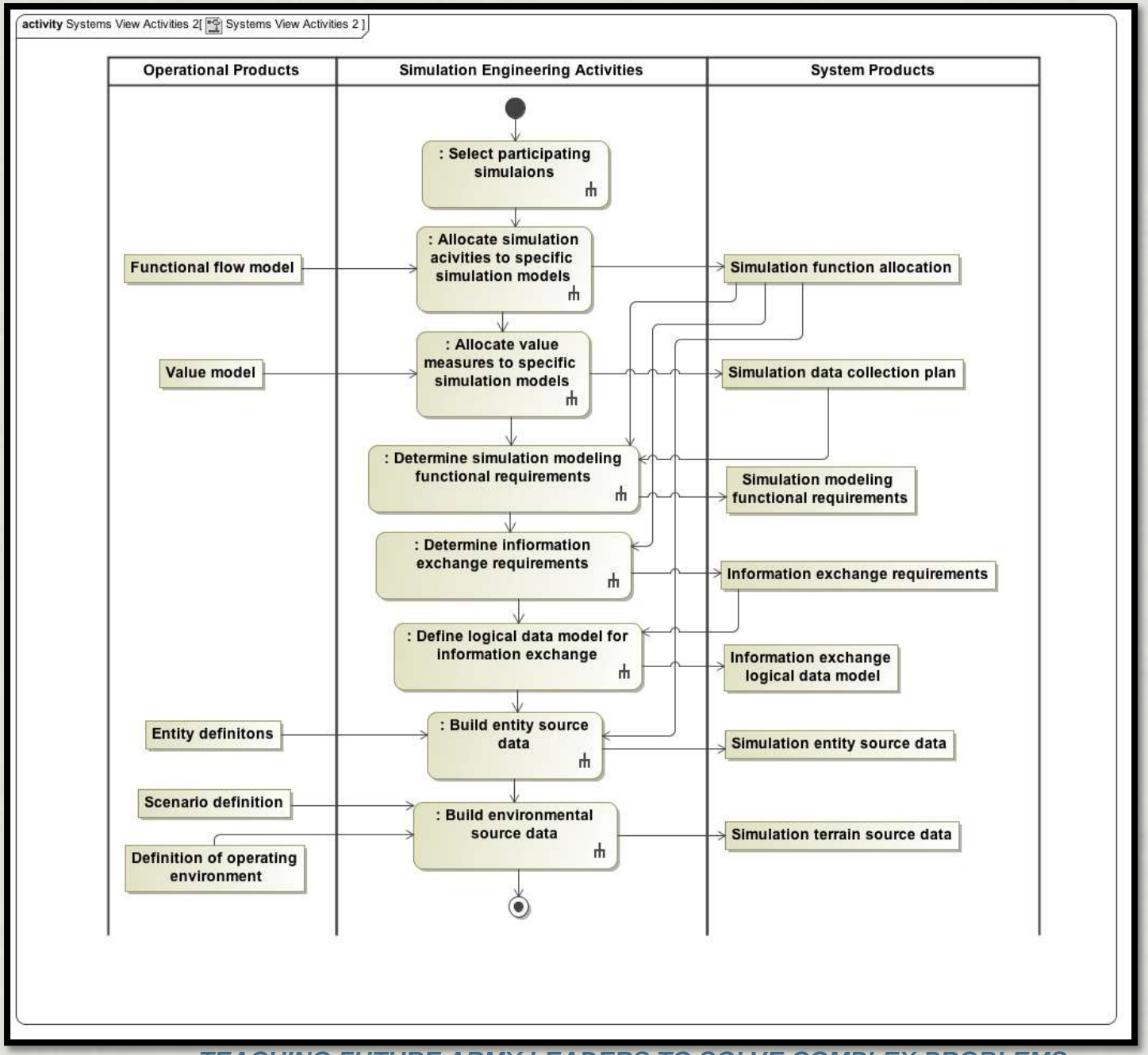


TEACHING FUTURE ARMY LEADERS TO SOLVE COMPLEX PROBLEMS

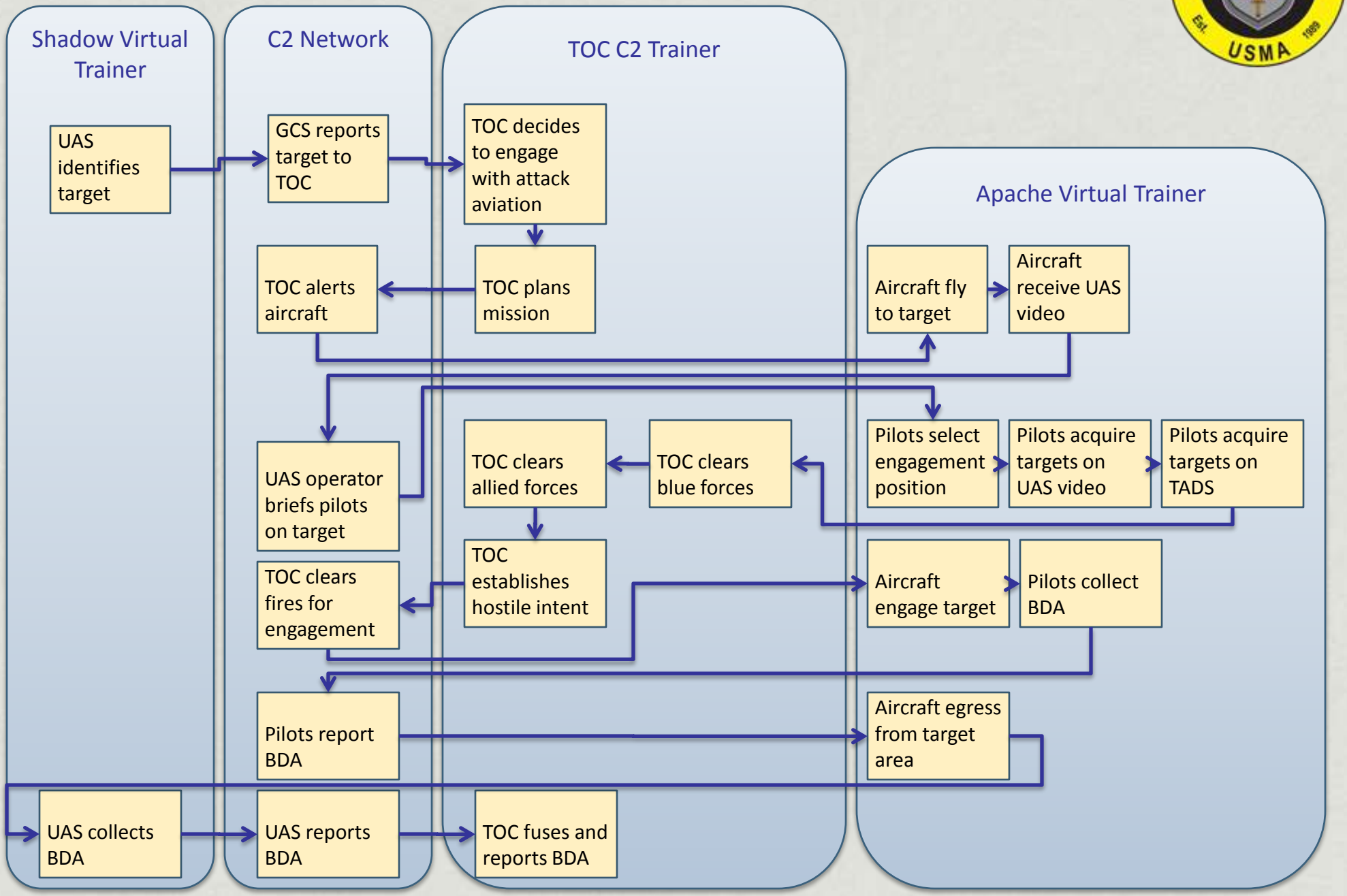
Standards-Based Scenario



Systems Engineering Process
Systems View Activities



Allocation of Mission Elements

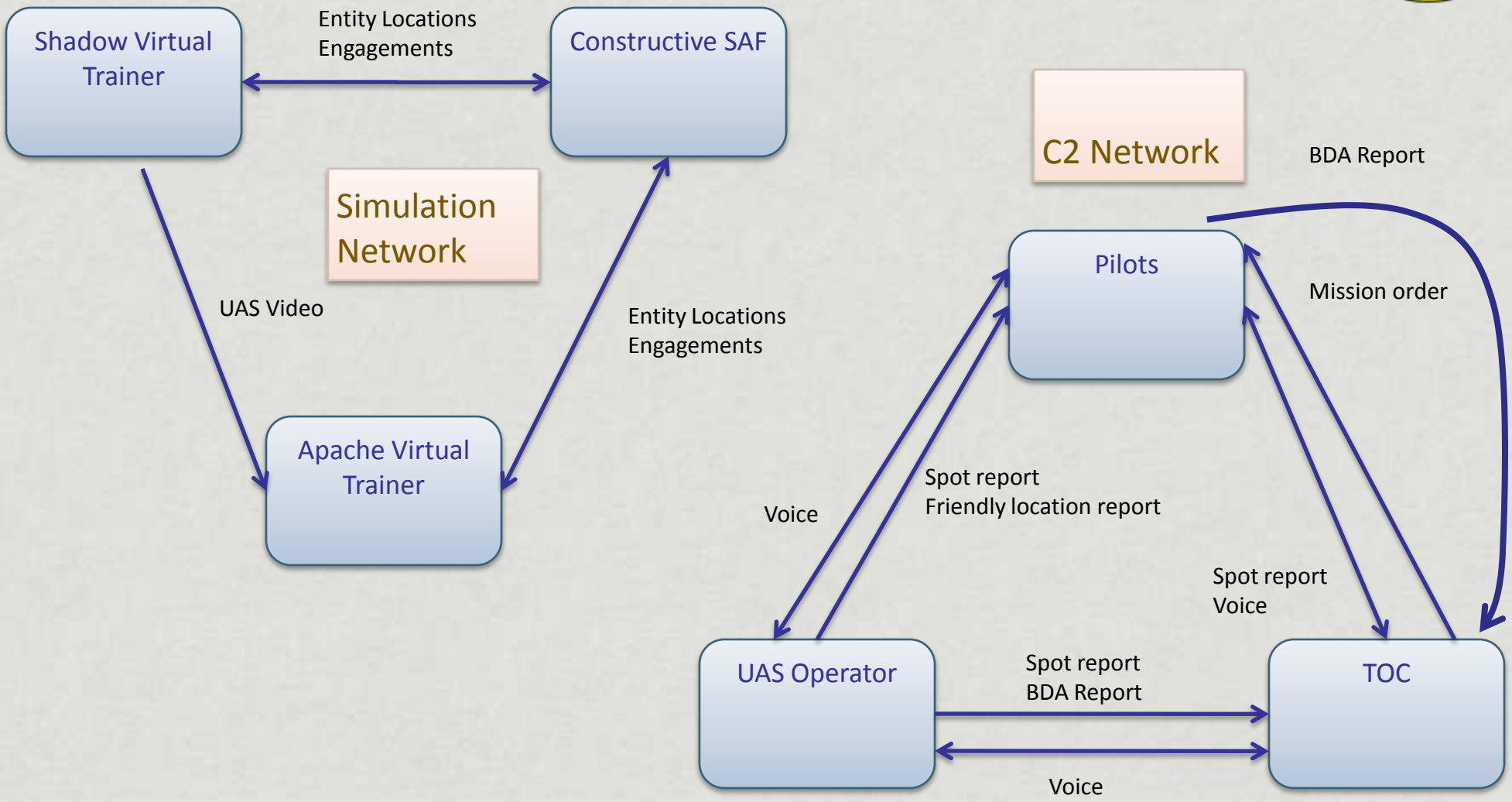


TEACHING FUTURE ARMY LEADERS TO SOLVE COMPLEX PROBLEMS

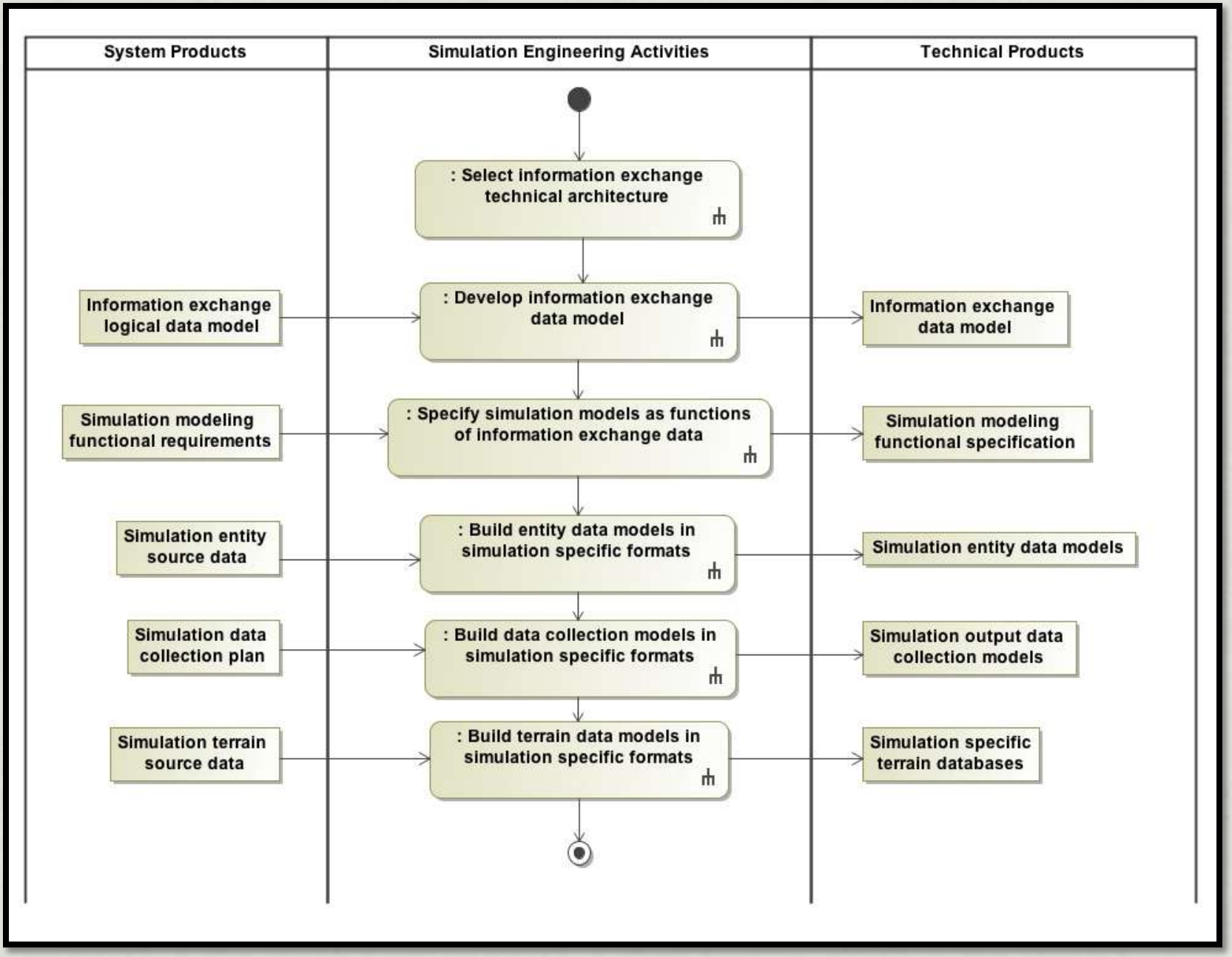
Mission Effectiveness Metrics

Shadow Virtual Trainer	C2 Network	TOC C2 Trainer	Apache Virtual Trainer	Constructive SAF	Simulation Network
	Time required to ID target	Time required to plan mission	Time required to fly to position	% enemy targets destroyed	
	% correct target identifications	Time required to clear blue	Time required to identify target	# of fratricide incidents	
	% enemy damage correctly reported	Time required to clear allies	% enemy targets identified	# of collateral casualties	
		Time required to establish hostile intent	% friendly vehicles identified		
					% neutral targets identified

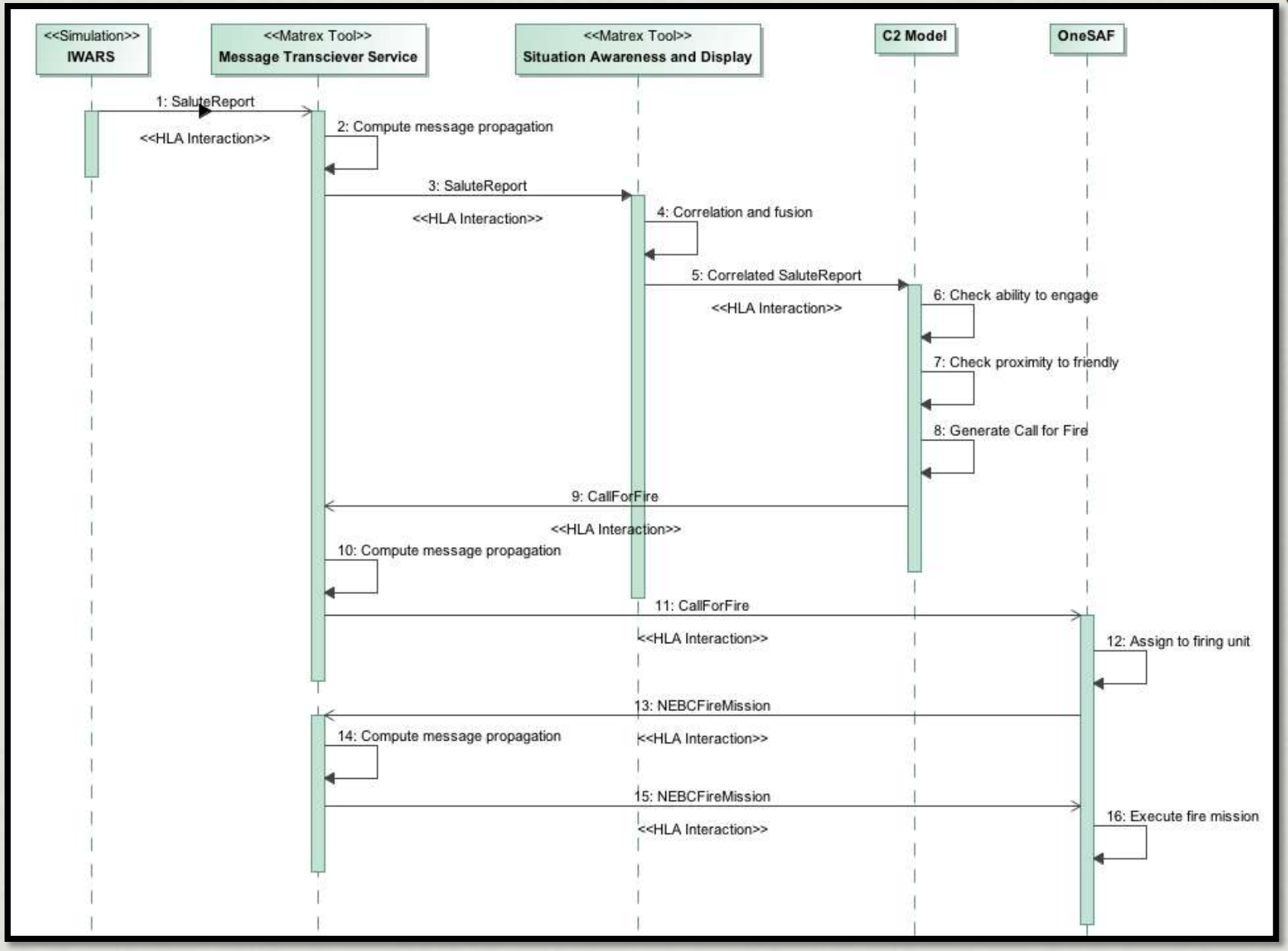
Communication Diagram



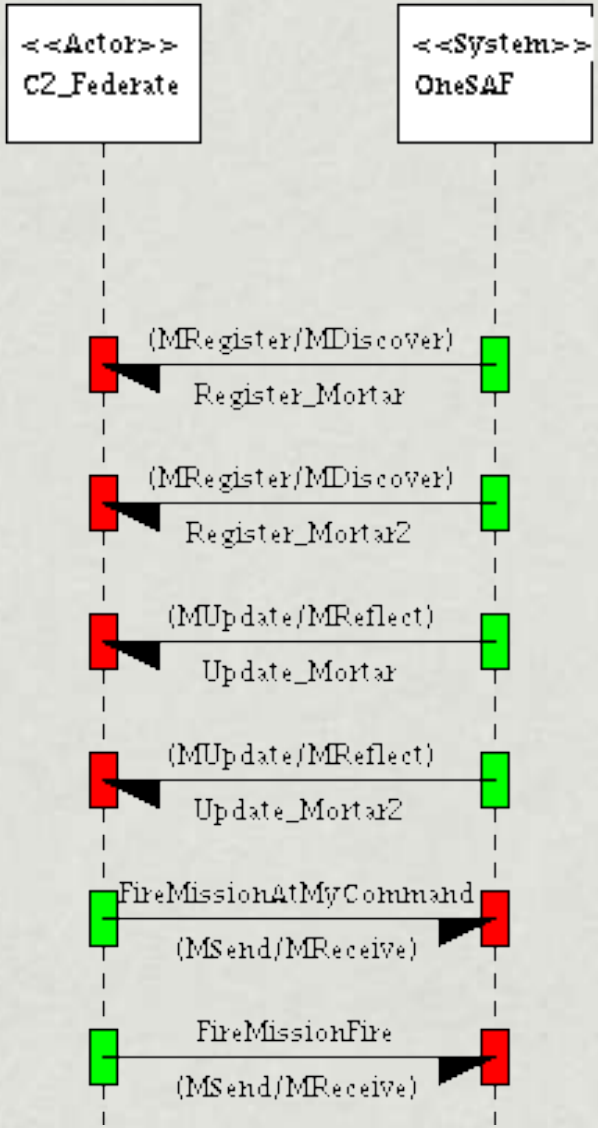
Systems Engineering Process
Technical View Activities



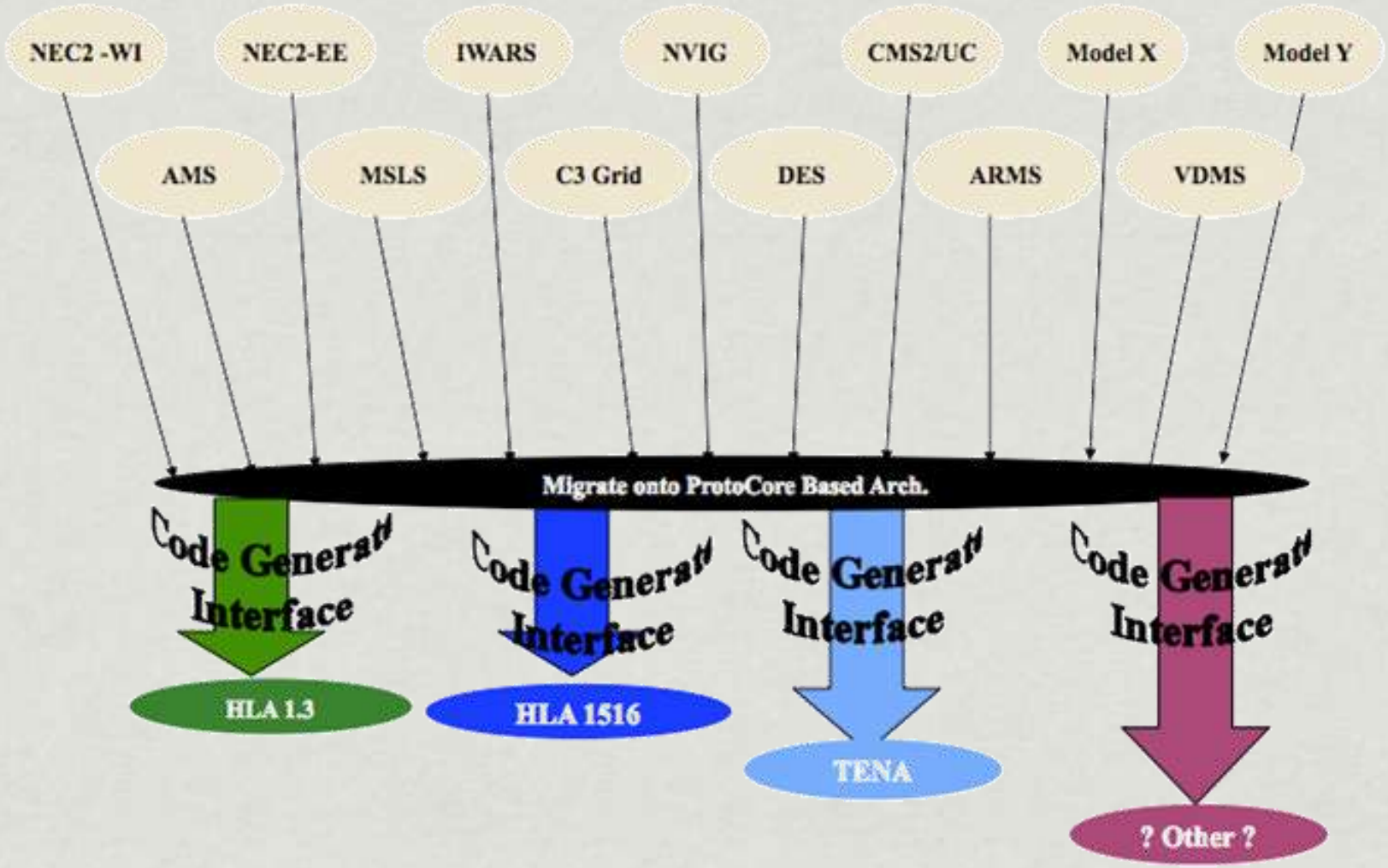
Technical View: Federation Design



Technical View
MATREX Test Cases



ProtoCore Code Generation Concept



AH64 Apache/UAS Cooperative Engagements

- Components
  - Deployable Virtual Training Environment – Cobra and UAS
  - Joint Semi-Automated Forces (JSAF)
  - One Semi-Automated Forces (OneSAF)
  - Joint Live Virtual Constructive Data Translator (JLVCDT)
- Combination of HLA and DIS
- Terrain generated with Terra Vista tools
- Entity mappings are a challenge
- SAF model has lower fidelity representation of character or vehicle actions

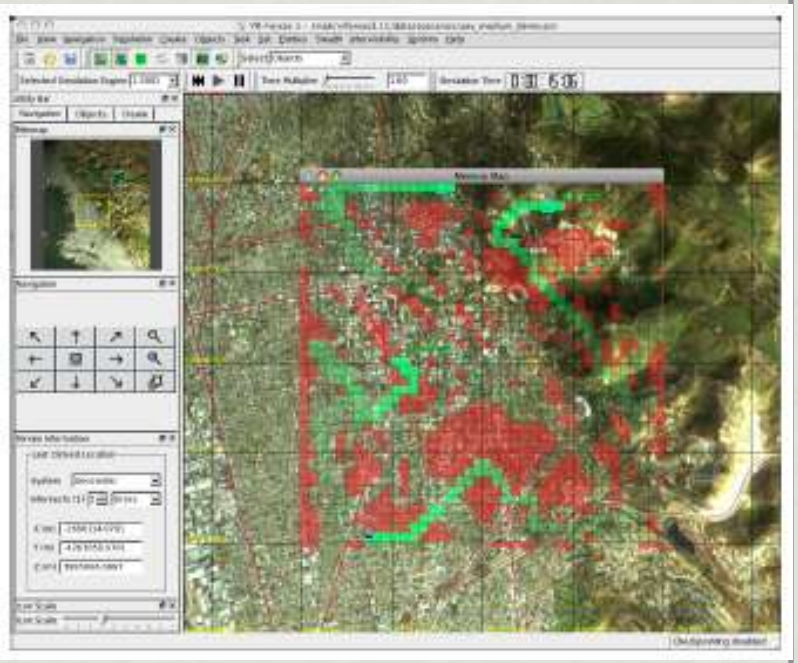
Soldier Command and Control

- Components
  - One Semi-Automated Forces (OneSAF) or COMBAT XXI
  - Infantry Warrior Simulation (IWARS)
  - MATREX Battle Command Management Services
  - Brigade and Below Propagation Protocol Communications Model

- HLA integration in MATREX architecture
- All models use OneSAF's Environmental Runtime Component (ERC)
- IWARS handles high fidelity soldier models
- Entity identification is a challenge
- Data collection using MATREX federation logger

Swarming Unmanned Aircraft Systems

- Components
  - VR-Forces SAF
  - Visualization federate (Mac-OS)
  - Swarming control federate
- DIS for information exchange



- Direct object integration for C2 and visualization
- Terrain representation in simulation and in GIS format for C2
- Custom data collection federate written

Systems Engineering Results

- Operational View
  - System functions to be simulated
  - Associated metrics to be collected via simulation
  - Operational descriptions of scenario, forces, and terrain
- Systems View
  - Allocation of system functions and metrics to supporting simulations
  - Simulation functional requirements
  - Simulation terrain and entity databases
  - Information exchange requirements and data model
- Technical View
  - Support for integration with specific interoperability protocol
  - Automatically generated test cases

Contacts

- LTC Robert Kewley, West Point Dept of Systems Engineering
  - (845) 938-5529
  - DSN 688-5529
  - Robert.Kewley@usma.edu
- URLs:
  - USMA: http://www.usma.edu/
  - Systems Engineering: http://portal.dean.usma.edu/departments/se/default.aspx
  - ORCEN: http://portal.dean.usma.edu/departments/se/Orcen/default.aspx